Knowl Inf Syst (2018) 55:305–331
https://doi.org/10.1007/s10115-017-1062-0

REGULAR PAPER

STEM: a suffix tree-based method for web data records extraction

Yixiang Fang[1] · Xiaoqin Xie[2] · Xiaofeng Zhang[3] · Reynold Cheng[1] · Zhiqiang Zhang[2]

Received: 15 April 2016 / Revised: 10 March 2017 / Accepted: 29 April 2017 /
Published online: 9 May 2017

Abstract To automatically extract data records from Web pages, the data record extraction algorithm is required to be robust and efficient. However, most of existing algorithms are not robust enough to cope with rich information or noisy data. In this paper, we propose a novel suffix tree-based extraction method (STEM) for this challenging task. First, we extract a sequence of identifiers from the tag paths of Web pages. Then, a suffix tree is built on top of this sequence and four refining filters are proposed to screen out data regions that might not contain data records. To evaluate model performance, we define an evaluation metric called pattern similarity and perform rigorous experiments on five real data sets. The promising experimental results have demonstrated that the proposed STEM is superior to the state-of-the-art algorithms like MDR, TPC and CTVS with respect to precision, recall and pattern similarity. Moreover, the time complexity of STEM is linear to the total number of HTML tags contained in Web pages, which indicates the potential applicability of STEM in a wide range of Web-scale data record extraction applications.

✉ Xiaofeng Zhang
zhangxiaofeng@hit.edu.cn

Yixiang Fang
yxfang@cs.hku.hk

Xiaoqin Xie
xiexiaoqin@hrbeu.edu.cn

Reynold Cheng
ckcheng@cs.hku.hk

Zhiqiang Zhang
zqzhang@hrbeu.edu.cn

1 Department of Computer Science, The University of Hong Kong, Pokfulam Road, Pokfulam, Hong Kong

2 College of Computer Science and Technology, Harbin Engineering University, Harbin, China

3 School of Computer Science, Harbin Institute of Technology Shenzhen Graduate School, G709, HIT Campus, University Town, Xili, Shenzhen, China

Keywords Web data extraction · Suffix tree · HTML tag path · Data Record pattern

1 Introduction

With the ever-increasing focus on big data analytics, collecting reliable data records, especially from *deep Web*, attracts a lot of research effort [1–4]. Generally, deep Web pages are dynamically generated in response to users' queries submitted to Web databases such as Google [5,6]. By wrapping related data records with predefined templates, cohesive data regions are naturally formed having similar visual appearances. Various algorithms are then proposed to discover data record patterns based on the similarity between visual appearances. However, visual appearances may not be regularly repeated due to the existence of rich information and noisy data such as decorative information which inevitably twists the underlying patterns, and consequently deteriorates the extraction accuracy of most of existing data record extraction methods.

In the literature, data record extraction algorithms can be classified into *automatic* approaches [5,7] and *semi-automatic* approaches [1]. For *automatic* approaches, they first treat aforementioned visually repeated data regions as page segments and then extract data records from these segments based on either tag information or attribute value. In these approaches, various pair-wise similarity metrics are designed to discover segments which are more likely to contain data records. However, these approaches are still far from perfect. This is because the underlying data record patterns are often not strictly repeated, due to the existence of three kinds of information, which greatly challenge the extraction methods. We summarize them as follows.

• *Rich information* Nowadays, Web pages generally are decorated with a lot of rich information to give a vivid illustration on data records for capturing users' attention. Consequently, data records in the query result pages often contain much rich information such as videos and images. For instance, Fig. 1a demonstrates the first query result page from Google for query keyword "Einstein", and each data record has four attributes, namely title, website, timestamp, and abstract. There are three data records: The first one only presents some relevant videos, images and a link; the second one gives the four attributes, an image and a link; and the third one only gives the four attributes. All the videos, images and extra links in the first and second records are considered as rich information, highlighted in the regions with red lines. It is obvious that the visual appearances of these data records are not regularly repeated. It is easy for *automatic* approaches such as [1–4] to extract records from Web pages without such rich information. However, they might not work well for Web pages like Fig. 1a.

• *Noisy data* In this paper, both the decorative tags and navigation bars separating data records are considered as noisy data. Figure 1b gives the query result page responded by Yahoo!. Each record has three attributes, namely title, website and abstract. The noisy data are highlighted in red rectangle, which separates the second and third records. Apparently, this gray line is a decorative tag, and its words "Ads related to: einstein" serve as a navigation bar.

• *Complex structure* Data records can have complex HTML structures. For example, the nested structures such as $<$ tr $>$ embedded in $<$ table $>$ are widely used to illustrate different attributes of data objects. However, aforementioned three kinds of information greatly twist the visual similarities of data records and therefore challenge most of existing data record extraction algorithms.

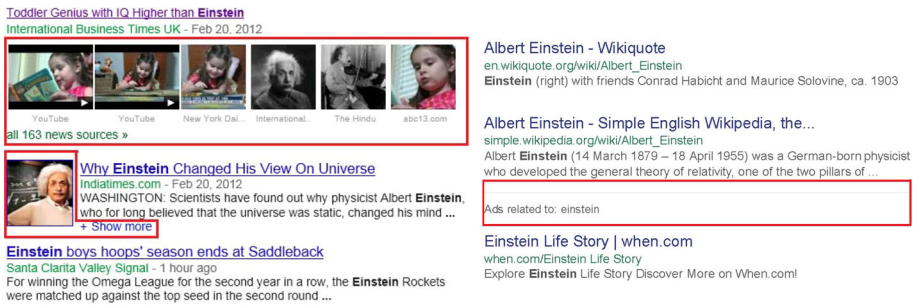


Fig. 1 Two example result pages of query keyword "Einstein" submitted to Google and Yahoo!. **a** Rich information, **b** noisy data

Thus, it is desired to propose a robust approach to extract data records from Web pages containing above information. In this paper, we focus on extracting multiple data records from single Web page, and propose a suffix tree-based data record extraction method (**STEM**). As is discussed, data records in a Web page generally follow the same data record pattern (DRP). To discover these DRPs, we first analyze the HTML tag paths and form a sequence of identifiers, called Web page sequence (WPS), in which each identifier corresponds to an HTML tag path. A suffix tree is then built on top of this WPS which is used to detect repeated subsequences from the WPS. Each repeated subsequence is assumed to contain data records wrapped in the same record patterns and thus is the target DRP. In order to extract DRPs efficiently, four refining filters are designed to screen out unnecessary nodes of the suffix tree. Finally, data records can be directly extracted using these filtered DRPs. At last, rigorous experiments are performed to evaluate the proposed approach. The promising results have demonstrated that STEM is superior to the state-of-the-art algorithms such as MDR [7], TPC [5] and CTVS [6] with respect to accuracy, recall and pattern similarity. Moreover, the time complexity of STEM is linear to the total number of HTML tags contained in a Web page and thus is more suitable for Web-scale applications.

The rest of this paper is organized as follows. We review related works in Sect. 2, formulate the problem in Sect. 3 and propose STEM in Sect. 4. Four refining filters are defined in Sect. 5, and we further discuss the STEM algorithm in Sect. 6. In Sect. 7, we present the experimental results and we conclude in Sect. 8.

2 Related work

Generally, approaches to extract data records from Web pages can be classified into twofold: *semi-automatic* and *automatic* approach. In this section, we will briefly review these related approaches. For more review of data extraction tools, systems, methods and applications, please refer to [1–4,8].

Early works for data extraction are mainly based on wrapper induction methods, and most of them call for manual interventions to define corresponding data extraction rules and thus can be seen as *semi-automatic* approaches. Liu et al. [9] developed an XML-enabled wrapper construction system for semi-automatic generation of wrapper programs. Gulhane et al. [10] developed a wrapper induction system called Vertex, which can extract structured data records from static Web pages. Etzioni et al. [11] exploited linguistic patterns to directly extract

Web data records. Recently, researchers believed that DOM trees of Web pages naturally organize Web pages into different page regions and underlying data records are generally contained in one or several such page regions. Alternatively, DOM tree-based approaches are then proposed [12,13]. Zheng et al. [14] proposed wrappers with a so-called broom structure which interleaves attribute value of tags with embedded data records. Dalvi et al. [15] presented a framework for the learning of wrappers from noisy data. However, as human intervention is often needed to extract the underlying DRPs, aforementioned approaches are not applicable in Web-scale applications.

Recently, more and more *automatic* approaches have been proposed which can be roughly classified into three categories: *DOM tree-based approaches, vision-based approaches* and *hybrid approaches*.

• *DOM tree-based approaches* These kinds of approaches generally define certain similarity measurement to locate data regions containing cohesive DRPs. The OMINI [16] applied a set of predefined heuristic rules to segment data regions. IEPAD [17] detected repeated substrings as token string to partition data regions and then extracted the corresponding data records. Similar approaches include string alignment [18], tree alignment [19], tree matching [20], equivalence classes [21], partial tree alignment [22], dynamic section extraction [23], and tree similarity-based algorithms [24].

Among these approaches, one of the most famous approaches is MDR [7]. MDR first segments Web pages into different regions and then compares the pair-wise distance between any two segments. Although MDR is superior to both OMINI and IEPAD, it fails to successfully extract data records when the deep Web pages contains noisy data such as advertisement and product comments. TPC [5] is then proposed which treats occurrence patterns of tag paths as visual signals, and the similarity between tag paths is calculated as the offset between visual signal vectors. Then, a spectral clustering algorithm is applied to find the dense clusters for the extraction of DRPs. However, TPC has several drawbacks. First, its extraction performance seriously relies on a set of pre-determined parameters, such as ε and the number of data regions. As Web pages vary greatly in different sites, it would inevitably result in the unstable model performance in terms of precision and recall. Second, TPC is susceptible to the noncontiguous DRPs. Once the data records contain nested structures like additional data attributes, TPC would separate one data region into several regions and thus incorrectly extract data records.

• *Vision-based approaches* As mentioned before, data records in the same region generally have similar visual appearances. Therefore, several studies are proposed to exploit visual features of data records to boost the extraction performance [25–27]. In [25], VENTex is proposed to identify page segments which are more possible to contain data records based on visual features of these page segments. Similarly, Cai et al. [26] proposed to partition Web pages using page layout features. Simon et al. [27] proposed to discover the repetitive patterns according to users' visual perception. However, as pointed out by [28], these visual feature-based approaches have two major limitations, i.e., the efficiency and the effectiveness in page rendering, due to the complicate design structures (e.g., CSS and Javascript) of Web pages.

• *Hybrid approaches* This kind of approaches combine the merit of DOM tree features and vision features-based approaches together to achieve a better data extraction performance. In [29], an approach called ViNTs is proposed which first utilizes the visual content (without HTML tags) to identify the regularities of page content and then combines them with the regularities of HTML tag structures to generate wrappers. In ViDE [30], visual features are mixed together with nonvisual information such as data types and symbols to build data record extractor. Similar work can be seen in [31]. Alternatively, there exist some research works

which extract data records based on ontology [32] or crowdsourcing [33,34]. Pasternack et al. [35] proposed a global optimization approach which uses maximum subsequence segmentation to extract articles from news websites. Weninger et al. [36,37] proposed to extract content of Web pages based on text-tag ratio calculated for each Web page. Furthermore, Sun et al. [38] combined the text-tag density of leaf nodes of the DOM tree to extract page contents. Wu et al. [39] extracted page contents by combing the DOM tree information with a machine learning-based approach. Song et al. [40] proposed a hybrid approach for content extraction by considering both text density and visual importance of DOM nodes. Although these works make use of structural information of the DOM tree, they can not directly extract data records merely based on the data record patterns discovered.

Recently, researchers have found that the DOM tree of each Web page can be modeled as a sequence of identifiers and the extraction of DRPs is equivalent to mining frequent subsequences from a set of sequence data [41–45]. However, existing sequence mining techniques cannot be directly applied for DRP extraction due to the following reasons. First, frequent subsequences are not necessarily contiguous, whereas DRPs are generally contiguous. Even for noncontiguous DRPs, their distance would be close enough if they were formed by similar data records. Second, if Web pages contain more than one DRP, then these DRPs may share common ancestor nodes even though they are not sibling nodes. To cope with these challenges, suffix tree is a natural choice as it is efficient to find repeated subsequences from a given sequence and is widely applied in many domains [46,47], such as indexing [42], document clustering [48,49] and topic detection [50]. Inspired by this, we propose a novel suffix tree-based method (STEM) for data record extraction in this paper. Note that an earlier version of this paper can be found in [45].

3 Problem formulation

In this section, we first introduce some related concepts, then formalize the record extraction problem and finally propose some useful lemmas.

3.1 Related concepts

Given a Web page as well as its DOM tree, an **HTML Tag Path** is a path from the root node[1] to a target node. The HTML tag path can be denoted as an ordered sequence of its ancestor nodes separated by "/". Note that each HTML tag has a unique path from its root node to itself. By traversing DOM tree in a pre-order manner, we can obtain the path of each HTML tag. An example is illustrated in Table 1. The left column lists the HTML code of each node, the second column shows the visiting order of each HTML tag, and the last column reports the corresponding HTML tag path of current node.

The **HTML Tag Path Identifier (HTPI)** of a Web page is an identifier for each unique HTML tag path. In this paper, HTPIs are represented by integers, and integers are assigned to HTML tag paths according to their appearing orders in a Web page. Table 2 lists the HTPI of each tag path. Note that the root node is assigned with 1 and the leaf node is assigned with a bigger value 7.

Definition 1 (*Web Page Sequence*) The **Web Page Sequence (WPS)** is an ordered sequence of HTPIs for a given Web page. The total number of HTPI in this sequence is called the length of WPS.

[1] Without further explanation, the DOM tree nodes mentioned in this paper are element nodes.

Table 1 HTML codes and HTML tag paths

HTML code	Order	HTML tag path
< html >	1	/html
< body >	2	/html/body
< div > An example	3	/html/body/div
< ul >	4	/html/body/div/ul
< li >	5	/html/body/div/ul/li
< a > Record 1 < /a >	6	/html/body/div/ul/li/a
< div > Details < /div >	7	/html/body/div/ul/li/div
< /li >	NA	NA
< li >	8	/html/body/div/ul/li
< a > Record 2 < /a >	9	/html/body/div/ul/li/a
< div > Details < /div >	10	/html/body/div/ul/li/div
< /li >< /ul >< /div >< /body >< /html >	NA	NA

Table 2 HTML tag paths and their unique identifiers

HTML tag path	HTPI
/html	1
/html/body	2
/html/body/div	3
/html/body/div/ul	4
/html/body/div/ul/li	5
/html/body/div/ul/li/a	6
/html/body/div/ul/li/div	7

For example, the WPS of the HTML page in Table 1 is (1, 2, 3, 4, 5, 6, 7, 5, 6, 7), and its length is 10. A data record contains at least one attribute of a data object (entity), and a data region is a page segment containing multiple data records. The **Data Record Pattern (DRP)** is defined as a list of HTML tag paths shared by multiple data records. Based on our previous discussions, a DRP can be represented by a sequence of HTPI. The total number of HTPIs in a DRP is called the length of DRP. In above example, there is one data region and its DRP is (5, 6, 7) with its pattern length equals to 3.

Definition 2 (*Web Page Suffix Tree*) The **Web Page Suffix Tree (WPST)** of a WPS is a compact trie tree containing all its suffix sequences. Each edge between nodes is labeled with a string.

To label outgoing edges of a node, the first character must be a distinct value. For a WPS with length n, the constructed WPST has at most n leaf nodes and $n - 1$ internal nodes. To build a WPST, we first insert all its suffix sequences into an empty tree and then merge their prefix sequences. Examples are plotted in Fig. 2a and b. Figure 2a plots the suffix sequences of a WPS (1, 2, 3, 4, 5, 6, 7, 5, 6, 7), and the WPST is plotted in Fig. 2b where circles denote the nodes. To simplify this figure, we omit those nodes whose labels are \$ and their ingoing edges.

Each node has two attributes: *node sequence* and *node frequency* which are defined as follows. The **Node Sequence** of a node q in a WPST is a subsequence of WPS which

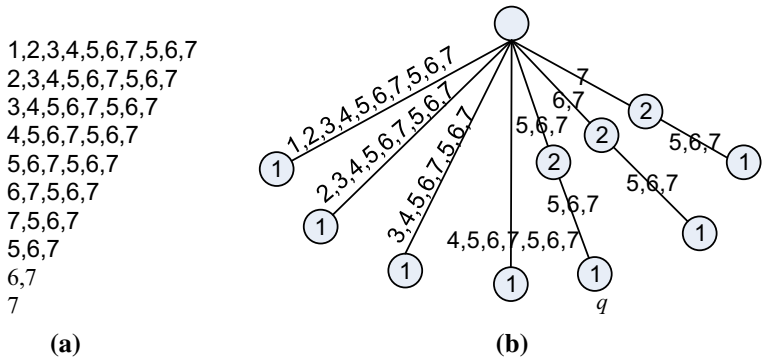


Fig. 2 An example Web page suffix tree. **a** Suffixes, **b** web page suffix tree

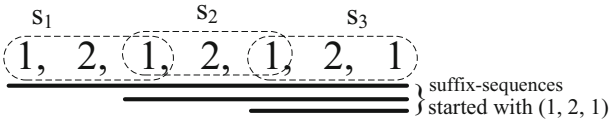


Fig. 3 An illustrating example for pattern frequency

is acquired by concatenating labels on its tag path. Each node sequence corresponds to a subsequence of WPS. To calculate node sequences, one can traverse the WPST from root node to the target node in a top-down manner, and thus the node sequence of q in Fig. 2b is (5, 6, 7, 5, 6, 7). Essentially, a node sequence is a candidate DRP, but only a few number of node sequences are the true DRPs. Thus, to extract DRPs from a Web page is equivalent to find particular subsequences from a given character string, i.e., WPS. This also implies that the task of Web data record extraction can be converted to a subsequence mining problem.

The **Node Frequency** $nf(q)$ of a node q is the number of suffix sequences, whose prefix sequences are node sequences of q. In Fig. 2b, the node frequencies of nodes are labeled in the circles. For example, the node frequency of q is 1, as there is only one suffix sequence whose prefix sequence is (5, 6, 7, 5, 6, 7). Given a WPS, the **Pattern Frequency** of a DRP is calculated as the maximum number of subsequences in the WPS. In Fig. 3, there are at most two subsequences, i.e., s_1 and s_3, can match suffix sequence (1, 2, 1) simultaneously, and thus the pattern frequency of (1, 2, 1) is 2.

3.2 Problem definition and proposed lemmas

By assuming that similar data records are usually grouped in the same data region, we summarize some key observations as follows:

- Any two different data regions in one Web page are non-overlapping, as they represent different groups of data records.
- Each data record contains at least two attributes, which implies that the corresponding DRP contains at least two pairs of HTML tags.
- Data records in the same data region are generally wrapped in the same template and thus are more likely to be embedded in the same DRPs formed by a set of repeated HTML tags.

Therefore, the data record extraction problem can be formulated as follows. Given an HTML source file f containing multiple data records, it is desired to directly extract underlying data records based on WPST built from Web pages.

Lemma 1 *Given a WPS wps and a DRP drp, drp's pattern frequency can be computed in a greedy manner:*
(a) Match drp with wps from left to right;
(b) Once a match is found, remove the matched part and its left identifiers from wps;
(c) Repeat Steps (a) and (b) until there is no more match found in wps. And the number of matches is the pattern frequency of drp.

Proof We will prove this lemma by contradiction. Suppose that there are t matches $mt_1, mt_2, \cdots, mt_t$ in wps ordered by their beginning position, and t is larger than the number of matches generated by the greedy algorithm. Then we perform the following processes: (1) If mt_1 is not the leftmost match of wps, we generate a new match, i.e., the leftmost match; otherwise, we just generate the same match as mt_1; (2) We remove the leftmost match and its left identifiers from wps; and (3) Repeat above procedures for the rest matches until there is no more match. It is easy to observe that for each match mt_i $(1 \leq i \leq t)$, we can always find a corresponding new match by above procedures. Therefore, the number of new matches is also t, which contradicts with the assumption that t is larger than the number of matches generated by the greedy algorithm. Hence, the pattern frequency can be computed by Lemma 1. □

Lemma 2 *Given a WPS wps, a DRP drp and a node q, if q's node sequence equals to drp, then drp's pattern frequency is not larger than $nf(q)$.*

Proof For any subsequence of wps that matches drp, there must exist a suffix sequence of wps begun with drp, so the number of matches is less than or equals to the number of suffix sequences begun with drp. Hence, Lemma 2 holds. □

In Fig. 3, if we assume that there is a drp (1, 2, 1), and a node q whose node sequence is also (1, 2, 1), then q's node frequency is 3, as its node sequence is the prefix sequences of three suffix sequences of wps. So drp's pattern frequency, i.e., 2, is less than q's node frequency.

Lemma 3 *Given a node q and its child nodes $q_1, q_2, \cdots, q_c$, we denote the node frequencies of q, $q_1, \cdots, q_c$ by $nf(q)$, $nf(q_1), \cdots, nf(q_c)$, respectively. We have $nf(q_i) \leq nf(q)$ and $\sum_{i=1}^{c} nf(q_i) \leq nf(q)$, where $1 \leq i \leq c$ and c is the number of q's child nodes.*

Proof A suffix tree is a compressed trie tree where suffix sequences of s with the same prefix sequences are grouped into the same subtree. As q is the parent node of q_i, the node sequence of q is the prefix sequence of q_i. This implies that if the node sequence of q_i appears in the suffix tree, then the node sequence of q must exist in the same suffix tree. Therefore, we have $nf(q_i) \leq nf(q)$ and $\sum_{i=1}^{c} nf(q_i) \leq nf(q)$. □

4 The STEM approach

In the literature, most of existing approaches [5–7] for data record extraction mainly involve two steps: identifying data regions and extracting data records. Instead of finding data regions,

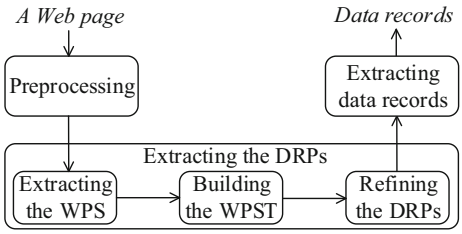


Fig. 4 The framework of the proposed STEM

the proposed STEM is to discover DRPs first and then extract data records using the DRPs. To discover the DRPs, we first represent a Web page by its Web page sequence. Then we build a suffix tree on this sequence. Finally, the repeated substrings in this suffix tree are extracted as DRPs. Once the DRPs are extracted, they will then be used to extract data records.

The overall framework of the proposed STEM is illustrated in Fig. 4. Steps to preprocess a Web page file f include HTML codes cleaning, repairing, etc. Then file f is parsed into a DOM tree, and all HTML tag paths are generated by traversing the DOM tree in a depth-first manner. By assigning an identifier to each path, we can acquire a WPS wps. A WPST st is then built on wps, and DRPs can be extracted from st using several proposed refining filters.

4.1 STEM algorithm

We illustrate STEM algorithm in Algorithm 1 and detail each main step as follows.

Step 1 Preprocessing the Web page file f

As there may exist page encoding or parsing errors, the source files of Web pages are not clean as: (1) some tags may lose their close tags; (2) some tags may mismatch with other close tags; and (3) some tags are incorrect or undefined. Therefore, necessary preprocessing is needed to clean Web pages. In our work, to repair tag losing or mismatching, we use JTidy,[2] a tool for parsing, repairing and outputting HTML codes in its normative format. All incorrect or undefined tags will be removed from f. To extract DRPs, we only focus on tags which are relevant to the Web page structure, and thus decorative tags such as $<$ em $>$, $<$ b $>$, $<$ u $>$ and $<$ i $>$ will be removed.

Step 2 Extracting the WPS wps

To convert a Web page into a WPS, we first parse the HTML codes into a DOM tree. By traversing the DOM tree in a depth-first manner, we can obtain the HTML tag path of each tag. A list of HTML tag paths is acquired by aggregating all tag paths. For each distinct path, we assign it with a unique identifier. As a consequence, a list of identifiers, i.e., wps, can be acquired.

Recall that the HTPIs of the DOM tree nodes are represented by integers. The DRP, i.e., a subsequence of wps, can also be represented by a list of integers. Thus, detecting DRPs is equivalent to check whether each subsequence of wps is a DRP or not. However, the time complexity of a naive approach which simply enumerates each subsequence will take $O(n^2)$. This is very time-consuming, especially when wps is very long. To improve the efficiency, we propose to use the suffix tree built on wps. As we will show later in Sect. 6, the time complexity of STEM is linear to the total number of tags contained in f.

[2] JTidy: http://sourceforge.net/projects/jtidy/?source=directory.

Algorithm 1 STEM Algorithm

```
1: procedure STEM(f) //f is a Web page
2:     preprocessing f; //Step 1
3:     extracting a WPS wps from f; //Step 2
4:     building a WPST st using wps; //Step 3
5:     drps ← FILTER(st) //Step 4
6:     for each drp in drps do
7:         extracting data records drs using drp; //Step 5
8:         outputting drs;
9: procedure FILTER(root)
10:    drpSet ← null, childList ← root.getChildNode();
11:    for each node in childList do
12:        if node cannot be pruned by any of the four filters then
13:            drpSet.add(node.nodeSequence);
14:        drpSet.add(FILTER(node));
15:    return drpSet;
```

Step 3 Building the WPST st

Given a wps, we can build a WPST st. A naive way to build st is to insert all suffix sequences of wps into an empty tree and merge prefix sequences of all suffix sequences. However, its time complexity is also $O(n^2)$. Fortunately, there exists linear time complexity suffix tree construction algorithms [51,52]. In the proposed STEM, we adopt a commonly used linear suffix tree construction algorithm Ukkonen [51] to build st. The node frequency can be computed simultaneously when we build the suffix tree.

Step 4 Extracting the DRPs $drps$

Recall that the node sequence of each node in st is a candidate DRP. To find true DRP $drps$, we need to filter out $drps$ that might not contain data records based on following observations: (1) the pattern length and frequency of a true pattern should be at least 2; (2) data pattern should be repeated in WPS; and (3) the tree structure of data pattern should be well kept. We then propose four refining filters, called frequency filter, gap filter, tree filter and repetition filter which are illustrated in Sect. 5. For each candidate DRP, we check whether it can be pruned by the four filters sequentially or not. If it cannot be pruned by any filter, then it is a true DRP.

Step 5 Extracting data records drs

DRPs generated from Step 4 will be used for data record extraction. Given a drp, we locate its beginning and last positions in wps. Suppose there are at most k subsequences which can match with drp, the wps can then be represented as follows:

$$wps = (s_0, drp, s_1, drp, s_2, \cdots, s_{k-1}, drp, s_k), \tag{1}$$

where s_i $(0 \leq i \leq k)$ is a subsequence, both $drps$ and s_i are called candidate sequences.

If there are no noisy data in data records, all s_i $(1 \leq i \leq k-1)$ will be null; otherwise, at least one of them is not null. To measure the closeness between s_i and drp, we define a record validation rate (RVR) for s_i, written as

$$rvr(s_i) = \frac{len(lcs(drp, s_i))}{len(drp)}, \tag{2}$$

where $len(\cdot)$ calculates the length of an input sequence, and $lcs(drp, s_i)$ denotes the longest common subsequence between drp and s_i, which can be computed using algorithm proposed in [53].

For a candidate sequence, a higher RVR value means a higher validation, i.e., noisy data is less. A threshold parameter α ($0 < \alpha \leq 1$) is introduced to determine whether noisy data records should be extracted or not. Empirically, α is set to 0.6 in this paper. And if $rvr(s_i) \geq \alpha$, we extract this data record; otherwise, we skip it. Detailed steps to extract data records are summarized as follows: (1) rewrite wps using Eq. (1); (2) compute the RVR values of s_i ($0 \leq i \leq k$) and select target sequences; and (3) extract data record as well as its attribute values from HTML tag paths.

5 Refining filters

5.1 Frequency filter

As the frequency of a DRP must be at least 2, any node whose pattern frequency is smaller than 2 will be pruned. According to Lemma 2, pattern frequency of any node is always smaller than or equal to its node frequency. Therefore, we need to filter nodes whose node frequency is smaller than 2. Similarly, the length of a DRP must be greater than 2, and nodes whose node sequence is less than 2 will be pruned. Illustrating examples are given in Figs. 5 and 6. Figure 5 plots a DOM tree of a Web page and the identifiers of "/html", "/html/div", "/html/div/p", "/html/div/a", and "/html/div/img" are assigned with 1, 2, 3, 4, and 5, respectively. Accordingly, its WPS is (1, 2, 3, 4, 2, 3, 4, 5, 2, 3, 4, 2, 3, 4, 2). In Fig. 6, nodes marked with red crosses will be pruned by the frequency filter. For example, node q_1 is pruned as its node frequency is 1 (<2), and q_2 is pruned as the length of its node sequence is also 1 (<2).

5.2 Gap filter

Generally, data records are contiguously distributed over data regions, which implies that the corresponding DRP is also contiguous in WPS. Recall that Eq. (1) aligns WPSs with candidate DRPs. Ideally, the gap sequences among *drps*, i.e., s_i ($1 \leq i \leq k$–1), are empty sequences. However, due to the existence of noisy data, some of them may not be empty. To prune noncontiguous sequences and allow the existence of some noisy data, we propose this gap filter. Given a node q with its node sequence s, whose length is l, the gap filter works as follows:

Step 1 For the sub-trees rooted at q, we traverse it in a depth-first manner. During the traversing, we concatenate the sequences traversed and stop traversing when the length of corresponding concatenated sequence exceeds l. Finally, a set S of nodes is collected as well as their corresponding concatenated sequences.

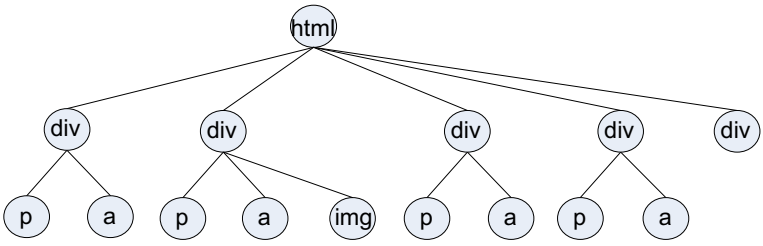


Fig. 5 A DOM tree

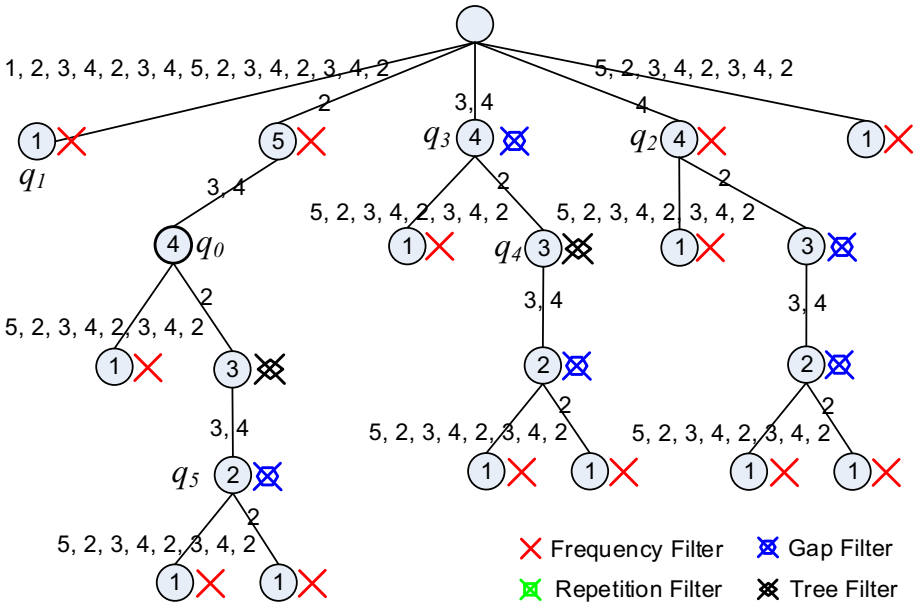


Fig. 6 An example of how refining filters work

Step 2 For each concatenated sequence with its corresponding node $q' \in S$, we need to find the subsequence s' which contains the first l HTPIs. Then, we can acquire $lcs(s, s')$ denoting the longest common subsequence between s and s'. For a DPR containing multiple data records, its visual appearances should be highly similar to each other as they are wrapped using the same template. According to Eq. (2), if

$$rvr(s) = \frac{len(lcs(s, s'))}{len(s)} \geq \alpha, \tag{3}$$

then s' is considered as a valid repetition of s and we output the node frequency of q. The summation of all such node frequencies is denoted as h.

Step 3 For a true DRP, we expect the proportion of contiguous records should be larger than a predefined threshold β, and we call it pattern validation rate (PVR), defined as:

$$pvr(q) = \frac{h}{pf(q)} \geq \beta, \tag{4}$$

where $pf(q)$ is the pattern frequency of q's node sequence. A higher PVR value implies a higher possibility that s could be a true DRP.

According to Lemma 2, we have $pf(q) \leq nf(q)$, and thus $\frac{h}{pf(q)} \geq \frac{h}{nf(q)}$. Hence, the gap filter can prune q, if

$$\frac{h}{nf(q)} \geq \beta. \tag{5}$$

Based on Lemma 3, we also have $h \leq nf(q)$ and $0 \leq \frac{h}{nf(q)} \leq 1$. Empirically, threshold β ($0 < \beta \leq 1$) is set to 0.6 in this paper.

The proposed gap filter allows the existence of noisy data among true DRPs. For example, after processing q_3 through step 1, two concatenated sequences (5, 2, 3, 4, 2, 3, 4, 2) and (2, 3, 4) are acquired which are reported in Fig. 6. In step 2, none of the subsequences (5, 2) and

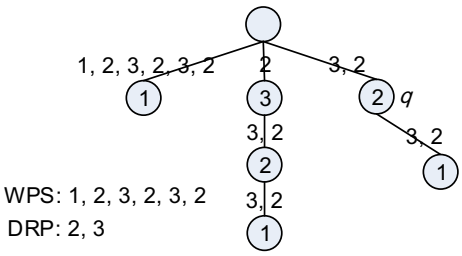


Fig. 7 An example illustrating the side effect of WPST

(2, 3) satisfies Eq. (3), which implies that (3, 4) is not contiguous one although it appears 4 times in the WPS. Then we can compute that $pvr(q_3) = 0$ and thus q_3 should be filtered. In contrast, for q_0 in Fig. 6, we can first obtain two concatenated sequences (5, 2, 3, 4, 2, 3, 4, 2) and (2, 3, 4) after step 1. In step 2, the computed node frequencies of (5, 2, 3) and (2, 3, 4) are 1 and 2, respectively. Accordingly, $pvr(q_0) = 0.75 \geq \beta$ and this node will not be filtered by gap filter.

5.3 Tree filter

As aforementioned, the DOM tree structure of a DRP should be well preserved. However, using suffix tree may detect repeated patterns whose tree structures are not well preserved. Figure 7 gives an example where the true DRP is (2, 3). However, we can only extract pattern (3, 2) using the previous filters. This is a side effect of using suffix tree to extract the DRPs. Fortunately, it only happens when a DRP repeats several times and terminates with one of its prefix. Due to the existence of the prefix, the order of HTPIs in DRP is rotated. In Fig. 7, the prefix of the DRP (2) is terminated when its subsequences are repeated. Thus, the true DRPs can be reconstructed by rotating the repeated patterns, and this is accomplished by the following tree filter.

As proposed in [7,27], the DOM tree structure of a DRP can only be one of the following cases.

- **Case 1:** The first tag is a root node, and other tags are its descendant nodes.
- **Case 2:** There are more than one tag serving as the root node, i.e., multiple root nodes. The parent nodes of these root nodes are the same node.

To check whether tag a is the parent node of tag b or not, we need to compare their tag paths: if a's path is the prefix of b's path, then a is b's ancestor node. For example in Fig. 5, the tag $<$ html $>$ is the parent node of tag $<$ div $>$, since "html" is the prefix of "html/div".

Specifically, for a node with node sequence s, the tree filter works as follows: (1) It first locates a list of HTML tag paths corresponding to identifiers in s; (2) Based on the paths, it identifies a list of root nodes by the following procedures: For each path p in the list, we check the relationship between each of its following paths p', and p. If p is a substring of p', we remove p' from the list; otherwise, we stop to check p and start to check the next tag path and the remaining paths are the paths of root nodes. Then the HTPI of the root node with the smallest HTPI is chosen. (3) It rotates s based on the first root node as follows:

$$(s_1, \cdots, s_{r-1}, s_r, \cdots, s_m) \xrightarrow{rotate} (s_r, \cdots, s_m, s_1, \cdots, s_{r-1}), \tag{6}$$

where s_r is the HTPI of the root node with the smallest HTPI. After rotation, if more than one s_r appears in s, then it eliminates the rest s_r. (4) Finally, it checks whether this rotated pattern has been extracted or not. The tree filter outputs this rotated pattern as a valid DRP if

it has not been extracted; otherwise, it is skipped. For example, in Fig. 7, the node sequence of node q can be rotated as a valid DRP (2, 3). However, in Fig. 6, the node sequence of node q_4 can be rotated as a valid pattern (2, 3, 4), but this pattern has been extracted from q_0, so it is filtered.

5.4 Repetition filter

Since data records are generated to represent a certain data object, at least one of its attributes should be distinctive. Repetition filter is proposed to filter nodes whose node sequences are the repetitions of their subsequences. Specifically, we consider two kinds of repetition: *simple repetition* and *nested repetition*. Given a candidate DRP, the repetition filter sequentially checks these two kinds of repetition with the following strategies.

Simple repetition Given a node q with its node sequence s, if s can be rewritten as

$$s = (s', s', \cdots, s'), \tag{7}$$

where s' is a subsequence of s and $|s| = l$, then s is a simple repetition of its subsequence s'. Since we aim to extract single data record, the repetition filter prunes q.

To check whether q can be pruned or not, a simple way is to check whether s is the repetition of any subsequence begun with the same identifier or not. However, its time complexity for checking a single node is $O(l^2)$, as we have to consider at most $l/2$ subsequences. To speedup this process, we turn to the structure of WPST. If s is the repetition of a true DRP s', then s' must have been detected as a DRP as there exists one ancestor node of p whose node sequence is s'. Therefore, to filter node q, we just need to enumerate each collected DRP and check whether its node sequence is the repetition of this DRP or not by using KMP algorithm [54]. For example, in Fig. 6, q_0 is checked before q_5 as q_0 is an ancestor node of q_5. In the refining process, we can detect that the node sequence of q_0 is a valid DRP as it can not be pruned by any filter.

Nested repetition Given a node q, if its tree structure can be represented by a list of repeated sub-trees, its node sequence is called nested repetition. This happens when data records are wrapped into a data region with several sub-regions. Consider the example illustrated in Fig. 8. There are 4 data records in the table, where records 1 and 2 are in one subregion, and

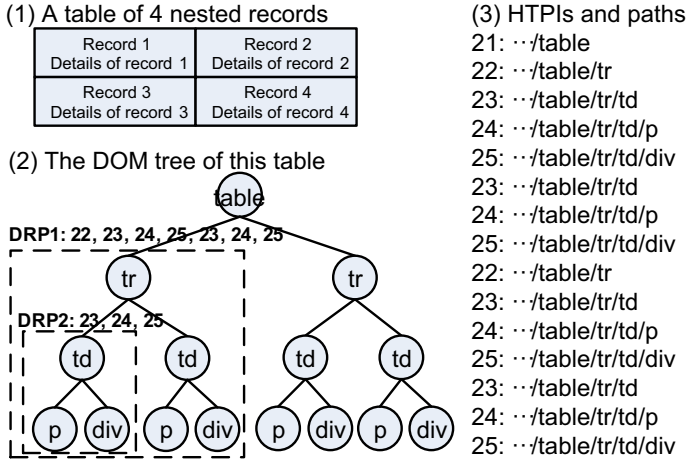


Fig. 8 An example detecting DRPs for nested records

records 3 and 4 are in another region. Their DOM tree structure is plotted in Fig. 8(2). By using our previous filters, we can extract two patterns: DRP1: (22, 23, 24, 25, 23, 24, 25) and DRP2: (23, 24, 25). Apparently, DRP1 is the nested repetition of DRP2, and it should be skipped. To prune a node with nested repetition, we first obtain the root nodes of its corresponding tree structure and then check whether the subsequence of its rest HTPIs is purely repetition or not. If does, we filter q; Otherwise, we keep it.

6 Discussions

In this section, we briefly discuss the proposed algorithm from two perspectives: time complexity analysis and quality of DRPs.

• *Time complexity analysis* Given a Web page f, let n denote the number of HTML tags in f, l ($2 \leq l \leq L$) denote the length of node sequence in WPST, and d denote the number of collected DRPs.

STEM has five steps. For the first two steps, preprocessing and extracting WPS, the complexity is $O(n)$. For the third step, as the length of the extracted WPS is at most n and we adopt an $O(n)$ algorithm to build the suffix tree, the time complexity of building WPST is also $O(n)$. For the fourth step to extract DRPs, we need to check whether a node should be pruned or not by the proposed filters. The corresponding analysis is given as follows. For frequency filter, the time complexity is $O(1)$ as we only need to check the node frequency and compare with its node sequence length. Since the WPST has at most $2n-1$ nodes and $n-1$ internal nodes, each internal node has 2 child nodes on the average. This implies that the average size of S is at most $2 \times l$ in gap filter. The total complexity of gap filter is $O(l^2)$, as its step 1, 2 and 3 can be completed in $O(l)$, $O(l^2)$ and $O(l)$, respectively. For tree filter, the complexity is $O(l^2)$. For repetition filter, since KMP algorithm can run linearly, the time complexity of this filter is $O(l \times d)$. Therefore, the time complexity of above four filters is $O(l^2 + l \times d)$. Since extracting data records using DRPs can be completed in $O(n \times l)$, the overall time complexity of refining DRPs is $O(n \times (l^2 + l \times d))$.

In practice, d and l are much smaller than n, and thus the overall time complexity approximates to $O(n)$. While the time complexities of TPC and CTVS are at least $O(n \times m + m^3)$ and $O(n \times l_1^2 \times l_2^2)$, where m is the number of unique tag paths, l_1 and l_2 are the maximum number of child nodes and the maximum length of a HTML tag path, respectively. In the worst case, both m, l_1 and l_2 are close to n. Therefore, it is apparent that the time complexity of STEM is smaller than that of TPC and CTVS.

• *Quality of DRPs* As the structures of extracted DRPs by STEM are kept very well, i.e., the HTML tag paths are ordered and complete, such patterns can be directly used for later data record extraction. While for other methods such as TPC, the extracted DRPs are generally disordered and incomplete.

Moreover, STEM is robust when extracting DRPs from Web pages with rich information and noisy data. The possible reasons are as follows. First, the gap filter of STEM assumes a given DRP is valid as long as the proportion of noisy data contained in this DRP are not higher than a predefined threshold. With this filter, more robust DRPs will be extracted. Second, according to the definition of WPS, STEM can easily extract DRPs from complicate data regions using simple sequence operations. In summary, STEM outperforms the state-of-the-art algorithms (e.g., TPC and CTVS) with respect to efficiency (i.e., time complexity) and effectiveness (quality of DRPs). Therefore, STEM is more suitable for large-scale data record extraction applications.

Table 3 Data sets used in experiments

Data set name	# of Websites	# of Webpages	# of Records
TB1	43	215	4,039
TB2	–	200	5,713
TB3	–	100	2,158
SD	15	75	766
MD	15	75	1,440

7 Experiments

For empirical evaluation, we have implemented STEM, TPC, and CTVS algorithms and evaluate their effectiveness in terms of precision, recall and pattern similarity. We implement all experiments in Java and run experiments on a PC with a 2.53 GHz CPU, 4GB memory and 500 GB disk.

7.1 Data sets

In our experiments, we use one data set collected by Yamada *et al.* [55], two data sets collected by Bing *et al.* [28], and two data sets collected by ourselves.[3] Table 3 summarizes some statistics of these data sets.[4]

The Yamada data set is chosen as the benchmark data set, and we call it TB1 for simplicity. This data set crawls 5 Web pages per each website out of total 43 websites and contains 215 Web pages. The true data records are already labeled in the data set for verifying the extraction accuracy. TB2 and TB3 are collected from online shopping and university websites. There are at most 2 and 5 Web pages collected from each website in TB2 and TB3, respectively. The main difference between them is that the Web pages of TB2 contain data records that are wrapped in regularly repeated patterns, while the Web pages of TB3 contain flat or intertwined flat data regions.

In addition, for correctness and robustness checking, we create two real data sets, i.e., search engine data set (**SD**) and miscellaneous data set (**MD**). For SD data set, we first choose 15 different popular search engines like Google, Baidu and etc. Then, we submit the same query to these search engines. From the query result pages, we only collect the top 5 Web pages. The MD data set contains much richer information like video, image, etc. To collect the MD data set, we first select 5 news websites, 5 e-commerce websites, and 5 multimedia websites. Then for each website, we query the Web pages and collect the first 5 returned Web pages.

We manually extract the data records in SD and MD data sets as the ground truth. Compared with TB1, TB2 and TB3, SD and MD contain much richer information and noisy data, especially for the first returning Web pages, which poses great challenges to the DRP extraction. For example, in the first returning Web page from Google, some records' HTML structures are very different with the DRP as shown in Fig. 1a.

[3] http://i.cs.hku.hk/~yxfang/public_data.rar.

[4] For TB2 and TB3, the number of websites are not specified in [28], and so we put "–" there.

7.2 Performance evaluation metrics

(1) *Precision* The precision of data record extraction is defined as

$$precision(f) = \frac{right(f)}{extract(f)} \tag{8}$$

where f denote a Web page, $extract(f)$ is the total number of data records extracted, and $right(f)$ is the total number of records extracted correctly. Note that when $extract(f)$=0, the $precision(f)$ is 0.

(2) *Recall* The recall is adopted to measure the extent that true data records have been extracted correctly, which is defined as

$$recall(f) = \frac{right(f)}{ground(f)} \tag{9}$$

where $ground(f)$ is the true number of data records in f.

(3) *Pattern similarity* It is proposed by us to measure the quality of extracted DRPs. This metric measures the similarity between the extracted pattern and the true DRP. Given a Web page, let drp, drp' denote the real DRP (ground truth) and extracted DRP, respectively. The corresponding similarity measurement can be defined as

$$sim(drp, drp') = \frac{len(drp) - ED(drp, drp')}{len(drp)} \tag{10}$$

where $len(drp)$, $ED(drp, drp')$ denote the length of drp and the edit distance [56] between drp and drp', respectively. The higher the value of $sim(drp, drp')$, the higher similarity between two compared patterns.

In addition, as discussed before, there may exist multiple data regions in one single page and not every data region contains desired data records. Generally, noisy data regions like advertisement and navigation contain less textual content to attract users' attention. And they are organized in a concise display format. Without loss of generality, we label those data regions containing more textual information as the target regions containing data records which are then used to evaluate extraction accuracy.

7.3 Performance evaluation

We first empirically investigate the effectiveness of the proposed four filters, evaluate the quality of extracted DRPs, and then examine the robustness of STEM on both SD and MD data sets, and finally compare STEM with the state-of-the-art algorithms. We performed preliminary studies on how the two threshold parameters β and α affect the validation rate of the extracted data records and DRPs. In this study, we found that when β and α are set to 0.6, the proposed approach can achieve the best performance. Thus, we set $\alpha, \beta = 0.6$ in the rest experiments.

7.3.1 Effectiveness of four refining filters

To demonstrate the effectiveness of the proposed refining filters, experiments are performed on all data sets. First, each Web page is preprocessed and then STEM is applied on each preprocessed page. The average number of total nodes, nodes filtered by frequency filter, gap filter, tree filter and repetition filter are recorded as well as the average number of remaining nodes.

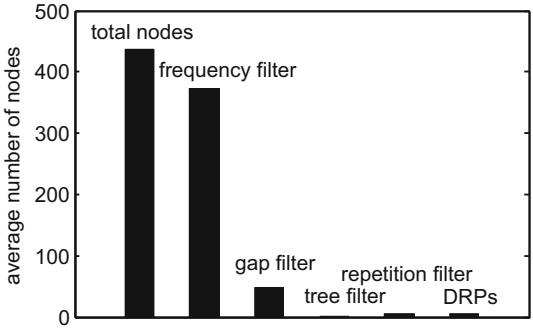


Fig. 9 The average number of nodes filtered by the refining filter

Figure 9 plots the average number of nodes in the WPSTs, the numbers of nodes pruned by different filters, and the number of remaining nodes, i.e., DRPs. The average number of total nodes is 436.4. Although the frequency filter is the simplest one, it filters around 373.9 nodes per WPST, which accounts for 86.8% (373.9/430.6) of total nodes filtered. In addition, the gap filter filters around 11.1% of total nodes filtered. It is noticed that the average number of remaining nodes (DRPs) is only 5.8, which is far less than that of total nodes. Thus, around 98.7% [(436.4–5.8)/436.4] of nodes are pruned by the filters, and this greatly speedups the overall data record extraction process.

7.3.2 The quality of DRPs

To evaluate the quality of the extracted DRPs, we choose the first result page of each search engine collected in SD and MD. We then extract DRPs manually from these 30 Web pages. We report the maximum length of the DRPs contained in each Web page in Fig. 10. It is well observed that the lengths of all the DRPs are less than 30. Note that L, the maximum DRP length, is set to 100.

We compare the quality of DRPs extracted by TPC and STEM by calculating their similarities with the ground truth DRPs based on Eq. (10). The similarities are plotted in Fig. 11. It is well noticed that for most of the testing Web pages, the similarity of DRPs extracted by STEM is much higher than that of the DRPs extracted by TPC. For 80% Web pages,

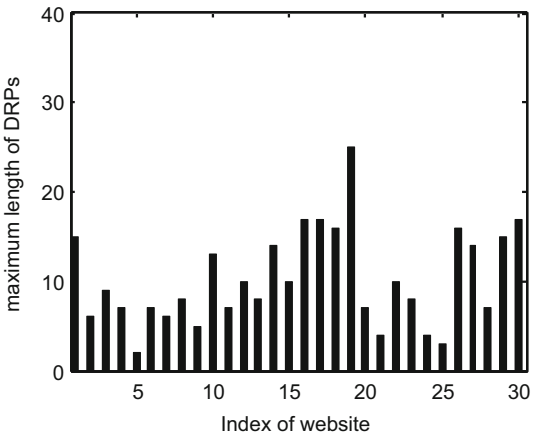


Fig. 10 DRP length

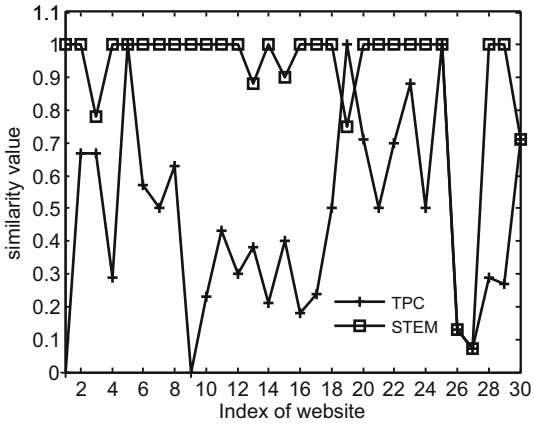


Fig. 11 Similarity comparison

the similarity of DRPs extracted by STEM is 1, whereas for 60% Web pages the similarities of DRPs extracted by STEM are smaller than 0.5. This indicates that when the data set contains much richer information or with complex display format, the quality of DRPs extracted by STEM is constantly good, whereas the performance of TPC is comparatively poor.

The reason that the performance of TPC is poor on data sets SD and MD lies in the existence of rich information and noisy data in data regions to be extracted. Recall that TPC detects the DRPS by clustering the visual signal vectors of HTPI. The *visual signal vector* of the i-th HTPI is a vector $s_i = (s_i(0), s_i(1), \cdots, s_i(n))$, where $s_i(j) = 1$ if the HTPI of this tag appears in the j-th position of a WPS, and $s_i(j) = 0$ otherwise; n denotes the length of WPS. Based on this definition, a Web page can be converted into a $m \times n$ matrix, where m and n denote the number of distinct HTML tag paths and the length of WPS, respectively. By assigning a bright pixel to "1" and a dark pixel to "0", we can acquire a visual representation of this Web page. Figure 12 depicts the visual signal vectors of HTIPs in the first query result page from "Baidu".

In Fig. 12, the white pixels in the rectangle and eclipses represent the true DRP and the noisy data in the real data regions, respectively. Table 4 shows the corresponding DRPs extracted by STEM and TPC. The true DRPs are (22, 23, 24, 31, 32, 38, 33), and the pattern length is 8. It is obvious that STEM can correctly extract the true DRPs from this kind of noisy Web pages, whereas TPC cannot. In Fig. 12, due to the existence of the rich information or noisy data, the similarity between vectors corresponding to (22, 23, 24, 31, 32, 38, 33) is very low, which leads to the poor clustering performance in TPC. As a result, TPC fails to extract DRPs. While for STEM, it is based on the occurrences of DRPs in WPS and thus it can extract the complete patterns.

7.3.3 Robustness analysis

To further demonstrate the robustness of STEM, we evaluate it on data sets SD and MD. The first query result page generally contains not only the well-structured textual contents but also the unstructured contents like url, image and video, as shown in Fig. 1. Even for the well-structured textual contents, the data records contained might not necessarily follow the same appearance pattern. In principle, this property of the first query result page requires a

Fig. 12 The visual signal vectors of the first query result page from "Baidu"

Table 4 DRPs extracted from the first query result page from "Baidu"

Method	HTPI	DRP (a list of HTML tag paths)
TPC	23	/html/body/div/div/div/table/tr
	31	/html/body/div/div/div/table/tr/td/h3
	32	/html/body/div/div/div/table/tr/td/h3/a
STEM	22	/html/body/div/div/div/table
	23	/html/body/div/div/div/table/tr
	24	/html/body/div/div/div/table/tr/td
	31	/html/body/div/div/div/table/tr/td/h3
	32	/html/body/div/div/div/table/tr/td/h3/a
	38	/html/body/div/div/div/table/tr/td/span
	33	/html/body/div/div/div/table/tr/td/a

more robust approach to acquire higher extraction accuracy. In the following experiments, we first extract the first query result page from these two data sets and then apply TPC and STEM on the extracted sub-dataset. In addition, we also compare the performance of TPC, CTVS and STEM on the first, second, third, fourth and fifth query result pages to evaluate their performance.

(1) *Comparison on the first query result page* The corresponding experimental results on data set SD and MD are depicted in Figs. 13 and 14, respectively. The true positive and false positive refer to the number of records which are extracted correctly and incorrectly, respectively, among the true data records. The true negative and false negative refer to the number of records which are extracted correctly and incorrectly, respectively, among the extracted data records.

From Fig. 13, it is apparent that the true-positive value of STEM is higher than that of TPC, and the false-positive value of STEM is less than that of TPC. Similar results are seen in Fig. 14. The reason why STEM outperforms TPC on this data set is as follows. TPC is easy to be affected by noisy data and thus fails to extract the DRPs correctly, which results in the increase in incorrect data records extracted. Moreover, the patterns extracted by TPC are incomplete due to the diverse display format of data records which in turn decreases the number of correct data records extracted.

(2) *Comparison on the i-th query result page* To perform this comparison, we first group all Web pages of data set SD and MD into 5 groups with each group only contains the corresponding ordinal number of query result pages, respectively. For example, the second

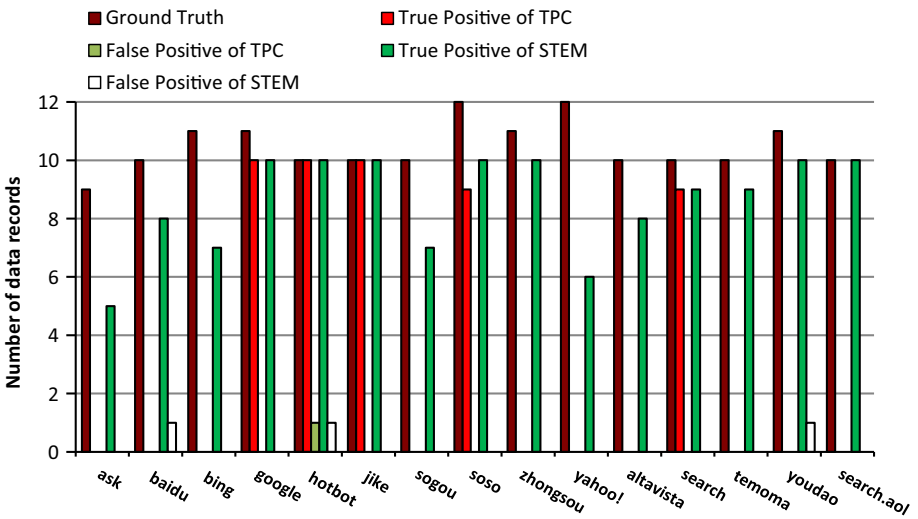


Fig. 13 Accuracy comparison between TPC and STEM on data set SD

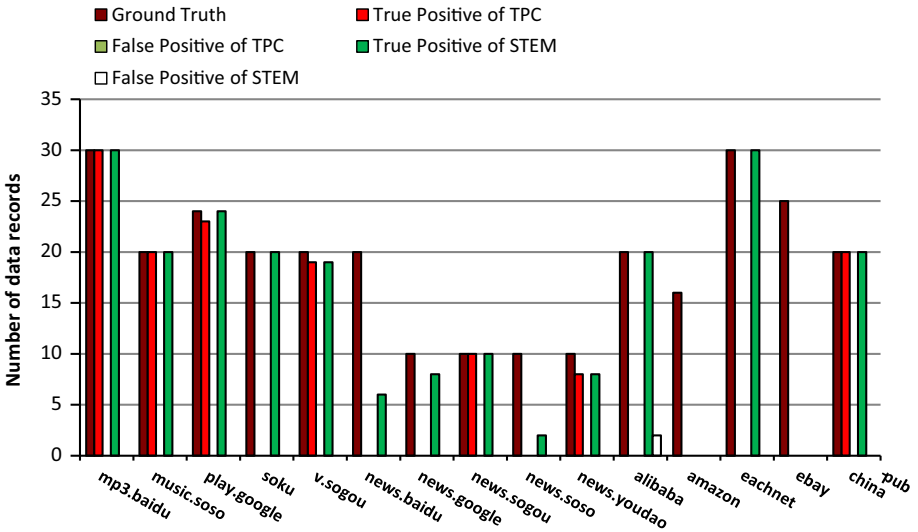


Fig. 14 Accuracy comparison between TPC and STEM on data set MD

group in SD contains all the second query result pages from SD. In each group, we apply TPC, CTVS and STEM and the average precision and recall are calculated and reported.

Figure 15 depicts the performance comparison of these three approaches. It is well noticed that STEM outperforms TPC and CTVS especially on group 1 in SD, and the precision and the recall of STEM is 65 and 48% higher than that of TPC, respectively. The overall precision and recall on group 1 in SD is very low (about 32.7% on precision and 30.4% on recall). We can also observe that both the precision and recall converge after the second group. The reason is that there are less rich information and noisy data in these data sets.

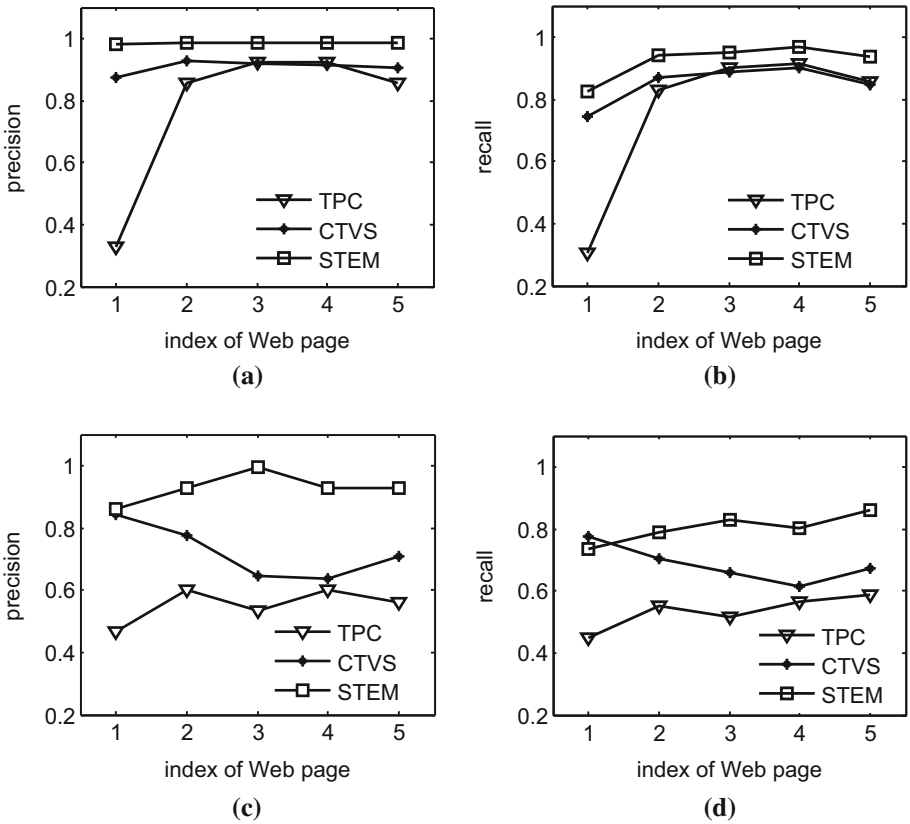


Fig. 15 Robustness performance comparison on data sets SD and MD. **a** Precision (SD), **b** recall (SD), **c** precision (MD), **d** recall (MD)

7.3.4 Data record extraction accuracy analysis

We also compare the Web data record extraction accuracy of MDR, TPC, CTVS and STEM. As the true data records are known, two commonly used evaluation metrics (e.g., precision and recall) are adopted in this experiment. Table 5 reports the average precision and recall of each algorithm. The results of MDR on TB1, TB2 and TB3 are from [5] and [28].

As reported in Table 5, STEM always achieves the best precision and recall on each data set. On data set TB1, the over performance of MDR is the poorest one. The precision of STEM is 34.9, 6.5 and 4.3% higher than that of MDR, TPC and CTVS, respectively. Similarly, the recall of STEM is 32.5, 3.0 and 5.7% higher than that of MDR, TPC and CTVS, respectively. Similar results could be found on TB2 and TB3. On the rest two data sets SD and MD, the performance of STEM also outperforms that of TPC and CTVS, especially on data set MD. For example, on data set MD, the precision of STEM is 36.7 and 22.1% higher than that of TPC and CTVS, and the recall of STEM is 27.0 and 13.1% higher than that of TPC and CTVS. This demonstrates that STEM is more robust than TPC and CTVS.

Table 5 The extraction accuracy comparison

Data set name	Algorithm	Precision	Recall
TB1	MDR	0.620	0.636
	TPC	0.904	0.931
	CTVS	0.926	0.904
	STEM	0.969	0.961
TB2	MDR	0.629	0.637
	TPC	0.792	0.795
	CTVS	0.865	0.854
	STEM	0.904	0.905
TB3	MDR	0.715	0.694
	TPC	0.810	0.815
	CTVS	0.919	0.941
	STEM	0.950	0.957
SD	TPC	0.776	0.754
	CTVS	0.909	0.850
	STEM	0.986	0.923
MD	TPC	0.560	0.533
	CTVS	0.707	0.672
	STEM	0.927	0.803

8 Conclusion and future work

In this paper, we propose a suffix tree-based extraction method (STEM) to extract data records from Web pages. STEM first extracts HTML tag paths from a Web page and transfers these paths into a Web page sequence, i.e., a list of identifiers. We find that only some specific subsequences of the WPS are related to the target DRPs. A suffix tree is then built on this WPS, and four refining filters are proposed to extract the DRPs. Finally, the extracted DRPs are applied to extract data records. Compared with the state-of-the-art algorithms such as MDR, TPC and CTVS, STEM is superior in terms of precision, recall and pattern similarity. Furthermore, STEM is demonstrated to be more robust if Web pages contain rich information, noisy data and complex structures, whereas the compared algorithms are susceptible to such Web pages. Among these algorithms, STEM is the most efficient one, whose time complexity is linearly to the total number of HTML tags contained in the Web page. This admirable property indicates the wide applicability of STEM in Web-scale applications where more reliable data records are expected to extract. Moreover, we are interested in extending STEM to extract data records embedded in dynamic script such as Javascript as this kind of data records generally does not follow the same data record pattern [45].

Acknowledgements Reynold Cheng and Yixiang Fang were supported by the Research Grants Council of Hong Kong (RGC Projects HKU 17229116 and 17205115) and the University of Hong Kong (Projects 102009508 and 104004129). Xiaoqin Xie was supported by the China Scholarship Council and the National Natural Science Foundation of China (Nos. 61202090, 61370084, 61272184), the Science and Technology Innovation Talents Special Fund of Harbin under Grant (No. 2015RQQXJ067), the Fundamental Research Funds for the Central Universities under Grant (No. HEUCF10060). Xiaofeng Zhang was partially supported by the National Science Foundation of China under Grant No. 61370213, National Key Technology Support Program of China under Grant No. 2014BAL05B06, and Shenzhen Science and Technology Program under Grant No. JCYJ20160330163900579. We would like to thank the reviewers for their insightful comments.

References

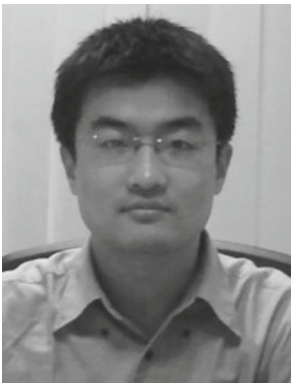
1. Laender A, Riberro-Neto B, Silva A, Teixira J (2002) A brief survey of web data extraction tools. In: SIGMOD record, vol 31(2). ACM, New York, pp 84–93
2. Chang C-H, Kayed M, Girgis MR, Shaala KF (2006) A survey of web information extraction systems. IEEE Trans Knowl Data Eng. 18(10):1411–1428
3. Sleiman H, Corchuelo R et al (2013) A survey on region extractors from web documents. IEEE Trans Knowl Data Eng 25(9):1960–1981
4. Ferrara E, De Meo P, Fiumara G, Baumgartner R (2014) Web data extraction, applications and techniques: a survey. Knowl Based Syst 70:301–323
5. Miao G, Tatemura J, Hsiung W-P, Sawires A, Moser L (2009) Extracting data records from the web using tag path clustering. In: Proceedings of the 18th international conference on World Wide Web (WWW). ACM, New York, pp 981–990
6. Su W, Wang J, Wang J, Lochovsky FH, Liu Y (2012) Combing tag and value similarity for data extraction and alignment. IEEE Trans Knowl Data Eng 24(7):1186–1200
7. Liu B, Grossman R, Zhai Y (2003) Mining data records in web pages. In: Proceedings of the ninth ACM SIGKDD international conference on knowledge discovery and data mining. ACM, New York, pp 601–606
8. Baumgartner R, Gottlob G, Herzog M (2009) Scalable web data extraction for online market intelligence. Proc VLDB Endow 2(2):1512–1523
9. Liu L, Pu C, Han W (2000) Xwrap: an xml-enabled wrapper construction system for web information sources. In: Proceedings of the IEEE 16th international conference on data engineering, pp 611–621
10. Gulhane P, Madaan A, Mehta R, et al (2011) Web-scale information extraction with vertex. In: ICDE, pp 209–220
11. Etzioni O, Cafarella M, Downey D, Kok S, Popescu A-M, Shaked T, Soderland S, Weld DS, Yates A (2004) Web-scale information extraction in knowitall (preliminary results). In: Proceedings of the 13th international conference on World Wide Web. ACM, New York, pp 100–110
12. Gupta S, Kaiser G, Neistadt D, Grimm P (2003) Dom-based content extraction of html documents. In: Proceedings of the 12th international conference on World Wide Web, pp 207–214
13. Gupta S, Kaiser GE, Grimm P, Chiang MF, Starren J (2005) Automating content extraction of html documents. In: Proceedings of the 14th international conference on World Wide Web, vol 8, no. 2, pp 179–224, 6
14. Zheng S, Song R, Wen J-R, Giles CL (2009) Efficient record-level wrapper induction. In: Proceedings of the 18th international conference on information and knowledge management (CIKM). ACM, New York, pp 47–56
15. Dalvi N, Kumar R, Soliman M (2011) Automatic wrappers for large scale web extraction. Proc VLDB Endow 4(4):219–230
16. Buttler D, Liu L, Pu C (2001) A fully automated object extraction system for the world wide web. In: Proceedings of IEEE 21st international conference on distributed computing systems, pp 361–370
17. Chang C, Lui S (2001) Iepad: information extraction based on pattern discovery. In: Proceedings of the 10th international conference on World Wide Web. ACM, New York, pp 681–688
18. Kayed M (2012) Peer matrix alignment: a new algorithm. In: Advances in knowledge discovery and data mining, pp 268–279
19. Zhai Y, Liu B (2005) Web data extraction based on partial tree alignment. In: Proceedings of the 14th international conference on World Wide Web (WWW). ACM, New York, pp 76–85
20. Liu B, Zhai Y (2005) Net-a system for extracting web data from flat and nested data records. In: WISE, vol 2005. Springer, pp 487–495
21. Arasu A, Garcia-Molina H (2003) Extracting structured data from web pages. In: Proceedings of ACM SIGMOD international conference on management of data, pp 337–348
22. Zhai Y, Liu B (2006) Structured data extraction from the web based on partial tree alignment. IEEE Trans Knowl Data Eng 18(12):1614–1628
23. Zhao H, Meng W, Yu C (2006) Automatic extraction of dynamic record sections from search engine result pages. In: Proceedings of the 32nd international conference on very large data bases, pp 989–1000
24. Bing L, Lam W, Gu Y (2011) Towards a unified solution: Data record region detection and segmentation. In: Proceedings of the 20th ACM international conference on information and knowledge management, pp 1265–1274
25. Gatterbauer W, Bohunsky P, Herzog M, Krüpl B, Pollak B (2007) Towards domain-independent information extraction from web tables. In: Proceedings of the 16th international conference on World Wide Web. ACM, New York, pp 71–80

26. Cai D, Yu S, Wen J-R, Ma W-Y (2003) Extracting content structure for web pages based on visual representation. In: Web technologies and applications, pp 406–417
27. Simon K, Lausen G (2005) Viper: augmenting automatic information extraction with visual perceptions. In: Proceedings of the 14th ACM international conference on information and knowledge management, pp 381–388
28. Bing L, Lam W, Wong T-L (2013) Robust detection of semi-structured web records using a dom structure-knowledge-driven model. ACM Trans Web 7(4):21:1–21:32
29. Zhao H, Meng W, Wu Z, Raghavan V, Yu C (2005) Fully automatic wrapper generation for search engines. In: Proceedings of the 14th international conference on World Wide Web (WWW). ACM, New York, pp 66–75
30. Liu W, Meng X, Meng W (2010) Vide: a vision-based approach for deep web data extraction. IEEE Trans Knowl Data Eng 22(3):447–460
31. Fumarola F, Weninger T, Barber R, Malerba D, Han J (2011) Hylien: a hybrid approach to general list extraction on the web. In: Proceedings of the 20th international conference on World Wide Web (WWW), pp 35–36
32. Furche T, Gottlob G, Grasso G, Guo X, Orsi G, Schallhart C, Wang C (2014) Diadem: thousands of websites to a single database. Proc VLDB Endow 7(14):1845–1856
33. Crescenzi V, Merialdo P, Qiu D (2013) Alfred: crowd assisted data extraction. In: Proceedings of the 22th international conference on World Wide Web (WWW), pp 297–300
34. Kondreddi SK, Triantafillou P, Weikum G (2014) Combining information extraction and human computing for crowdsourced knowledge acquisition. In: IEEE 30th international conference on data engineering, pp 988–999
35. Pasternack J, Roth D (2009) Extracting article text from the web with maximum subsequence segmentation. In: Proceedings of the 18th international conference on World Wide Web (WWW), pp 971–980
36. Weninger T, Hsu WH (2008) Text extraction from the web via text-to-tag ratio. In: 19th international workshop on database and expert systems application, pp 23–28
37. Weninger T, Hsu WH, Han J (2010) Cetr: content extraction via tag ratios. In: WWW, pp 971–980
38. Sun F, Song D, Liao L (2011) Dom based content extraction via text density. In: SIGIR, pp 245–254
39. Wu S, Liu J, Fan J (2015) Automatic web content extraction by combination of learning and grouping. In: Proceedings of the 24th international conference on World Wide Web (WWW), pp 1264–1274
40. Song D, Sun F, Liao L (2015) A hybrid approach for content extraction with text density and visual importance of dom nodes. Knowl Inf Syst 42(1):75–96
41. Chiu D-Y, Wu Y-H, Chen A (2009) Efficient frequent sequence mining by a dynamic strategy switching algorithm. VLDB J 18(1):303–327
42. Loh W-K, Ahn H (2011) A storage-efficient suffix tree construction algorithm for human genome sequences. IEICE Trans 94–D(12):2557–2560
43. Pei J, Han J, Mortazavi-Asl B, Pinto H, Chen Q, Dayal U, Hsu M-C (2001) Prefixspan: mining sequential patterns efficiently by prefix-projected pattern growth. In: ICCN, pp 215–224
44. Pei J, Han J, Mortazavi-Asl B, Wang J, Pinto H, Chen Q, Dayal U, Hsu M-C (2004) Mining sequential patterns by pattern-growth: the prefixspan approach. IEEE Trans Knowl Data Eng, vol 16
45. Xie X, Fang Y, Zhang Z, Li L (2012) Extracting data records from web using suffix tree. In: Proceedings of the 18th SIGKDD workshop on mining data semantics. ACM
46. Maaß M (1999) Suffix trees and their applications. Ferienakademie
47. Grossi R, Italiano GF (1993) Suffix trees and their applications in string algorithms. In: Proceedings of the 1st south American workshop on string processing, pp 57–76
48. Zamir O, Etzioni O (1998) Web document clustering: a feasibility demonstration. In: Proceedings of the 21st annual international ACM SIGIR conference on research and development in information retrieval, pp 46–54
49. Chim H, Deng X (2007) A new suffix tree similarity measure for document clustering. In Proceedings of the 16th international conference on World Wide Web. ACM, pp 121–130
50. Fang Y, Zhang H, Ye Y, Li X (2014) Detecting hot topics from twitter: a multiview approach. J Inf Sci 40(5):578–593
51. Ukkonen E (1995) On-line construction of suffix trees. Algorithmica 14(3):249–260
52. Farach M (1997) Optimal suffix tree construction with large alphabets. In: FOCS, pp 137–143
53. Greenberg RI (2003) Bounds on the number of longest common subsequences. CoRR, vol. cs.DM/0301030. [Online]. Available: http://arxiv.org/abs/cs.DM/0301030
54. Knuth DE, Morris JH, Pratt VR (1977) Fast pattern matching in strings. SAIM J Comput 6(2):323–350

55. Yamada Y, Craswell N, Nakatoh T, Hirokawa S (2004) Testbed for information extraction from deep web. In: Proceedings of the 13th international conference on World Wide Web (WWW). ACM, New York, pp 346–347
56. Navarro G (2001) A guided tour to approximate string matching. ACM Comput Surv 33(1):31–88

Yixiang Fang received the B.Eng. and M.S. degrees from Harbin Engineering University and ShenZhen Graduate School in Harbin Institute of Technology in 2010 and 2013, respectively. He is currently working towards the Ph.D. degree in Department of Computer Science in the University of Hong Kong (HKU) under the supervision of Dr. Reynold Cheng. His research interests mainly focus on big data analytics on spatiotemporal databases, graph databases, uncertain databases, and Web data mining.

Xiaoqin Xie received the Ph.D. degree from Tsinghua University in China in 2005. She is now an associate professor at college of computer science and technology of Harbin Engineering University. Her research interests include Web data mining, social network analyzing and mining, machine learning and service-oriented computing.



Xiaofeng Zhang received the MSc degree from Harbin Institute of Technology in 1999 and the Ph.D. degree from Hong Kong Baptist University in 2008. He has worked in R&D center of Peking University Founder Group and E-business Technology Institute of Hong Kong University. He is now an associate professor at department of computer science of Harbin Institute of Technology Shenzhen Graduate School. His research interests include data mining, machine learning and graph mining.

Reynold Cheng is an Associate Professor in the Department of Computer Science of the University of Hong Kong (HKU). He obtained his PhD from Department of Computer Science of Purdue University in 2005. Dr. Cheng was granted an Outstanding Young Researcher Award 2011-12 by HKU. He has served as PC members and reviewers for international conferences (e.g., SIGMOD, VLDB, ICDE, EDBT, KDD and ICDM) and journals (e.g., TKDE, TODS, VLDBJ and IS).

Zhiqiang Zhang born in 1973. He is a professor of Harbin Engineering University and received Ph.D degree in computer science from Tsinghua University, China. He is senior member of China Computer Federation. His main research interests include information retrieval, database, and intelligent information processing.